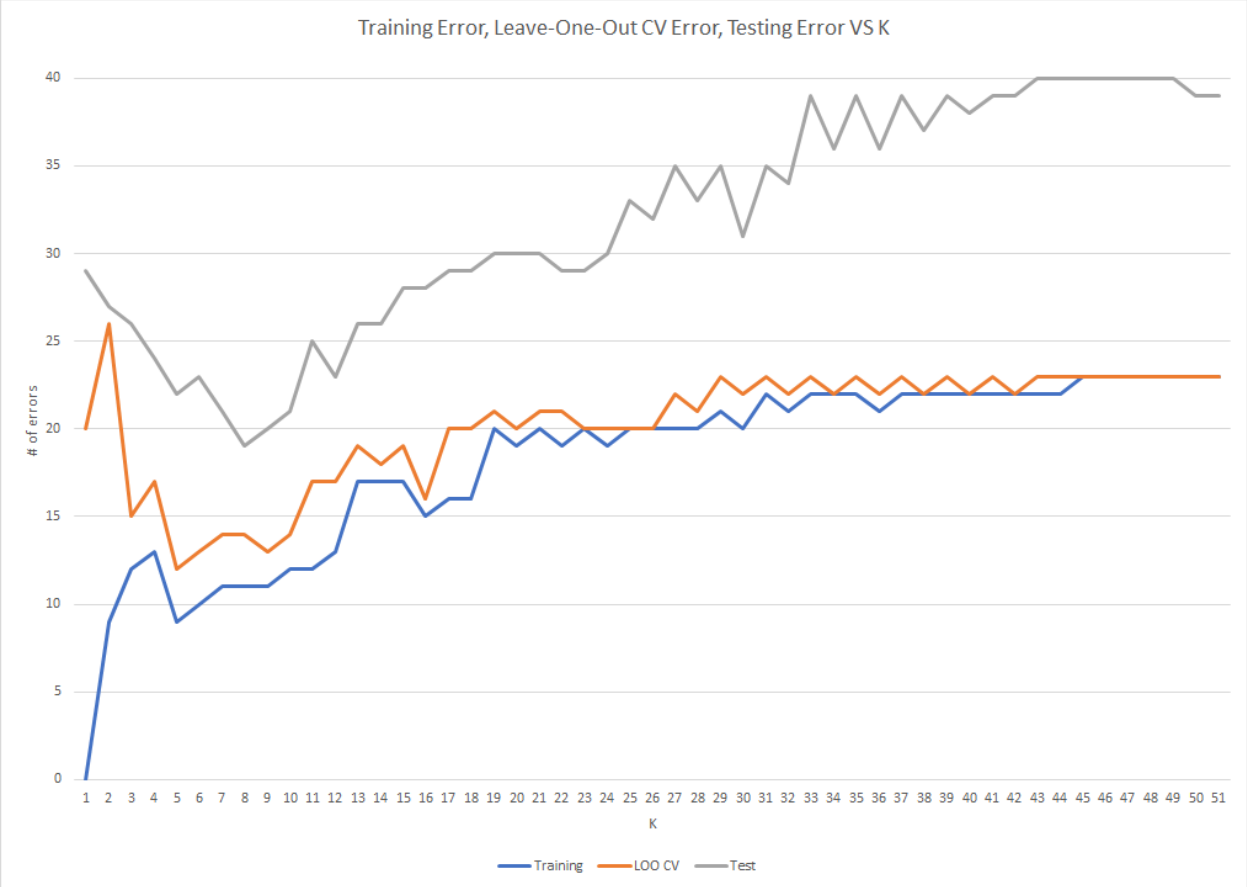
Logan Wingard
Daniel Ross

Implementations assignment 2

Part 1: KNN

1.1) See KNN.py of submission.
`python KNN.py` will run the KNN Training error, Leave-one-out CV, and Testing error over K=1 through K=51
1.2)



1.3)The absolute minimum in training error and LOO CV match the local minimum of the testing data at K=5. The Testing error is lowest at K=8 while the Training Error and LOO CV are only 2 errors greater than their minimums. Since the testing error is at a minimum here K=8 would be my choice for the KNN algorithm.

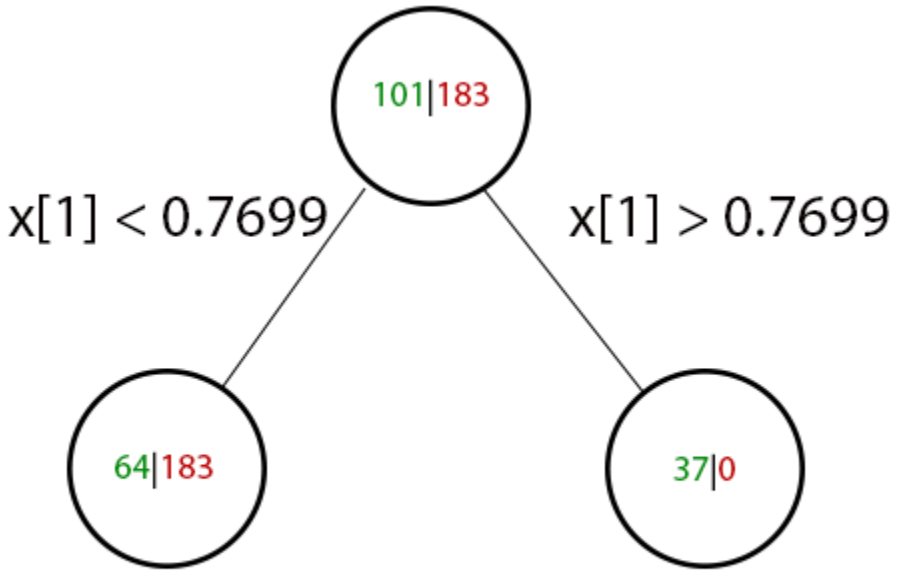
Part 2: Decision Tree

To run, just use the command:

```
python dt.py [depth]
```

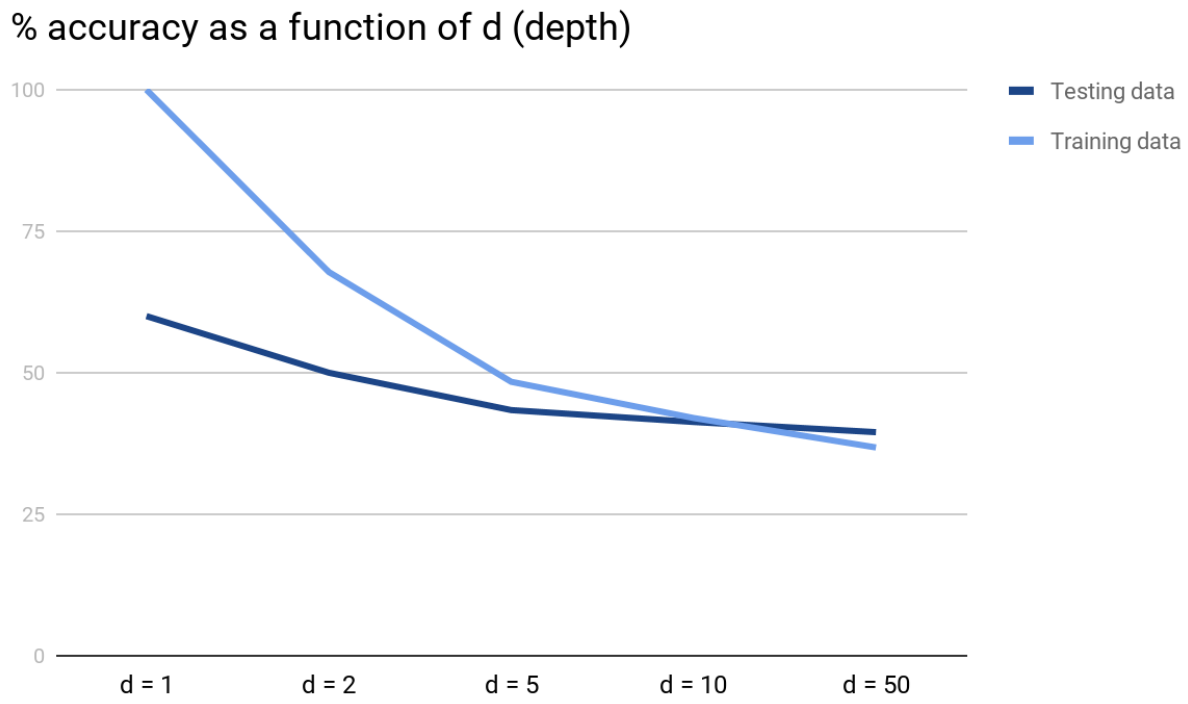
The depth argument is optional and defaults to 5.
The output you see will have the accuracy of the decision stump from part 1 on both testing and training data, as well as accuracy from the decision tree of depth d on both testing and training data.

1.
    1)



    2) The max gain after testing over the 284 tests(but only when y changes) over all 30 attributes was on attribute 1 with a gain of 0.1027.
    3) The accuracy for over testing on kss_test.csv data set is 61.27%
    And the accuracy on he training data set was 64.79%
    Making the errors test: 38.73% | train: 35.21

2.



The obvious observation of this data is that as the depth increases, the accuracy decreases. That isn't what should happen. Through my testing, I am pretty certain that the problem I am having is a syntax problem in my code, and the core algorithm I am implementing is correct. That being said, I will discuss the algorithm and why it should be increasing as d increases.

To determine the threshold, first I determined the information gain with the equation:

Gain = entropy(parent)-[average entropy(children)]

All I have to do is run this over every instance of data where y changes (post sort) and take the maximum value of all these gains. The threshold is the x value at the point of max information gain. For d (the depth of the decision tree) I find the best attribute in the set and split it at the threshold, then do the same with these subsets, decrementing d each time, until we can classify whether it is benign or not from the data in the decision tree. As d increases, more and more data is being taken into account which should increase the accuracy greatly as d increased.

Part 3: Integrating DT into KNN

3.1) When a particular example is run through KNN it's K neighbors are identified. Then each of the neighbors submits a vote as to what the example will be classified as. DT could be used in the voting process. Each K neighbors could have it's vote negated based on a distance threshold. This provides another way to limit the influence of superfluous neighbors in the calculation. Ideally this would decrease the rate at which the algorithm's accuracy decays as K approaches the full dataset.